FAT-Pointer based range TLB

Akilan Selvacoumar[1], Robert Stewart[1], Hans-Wolfgang Loidl[1], and Ryad Soobhany[1]

[1]Heriot-Watt University, UK

Abstract

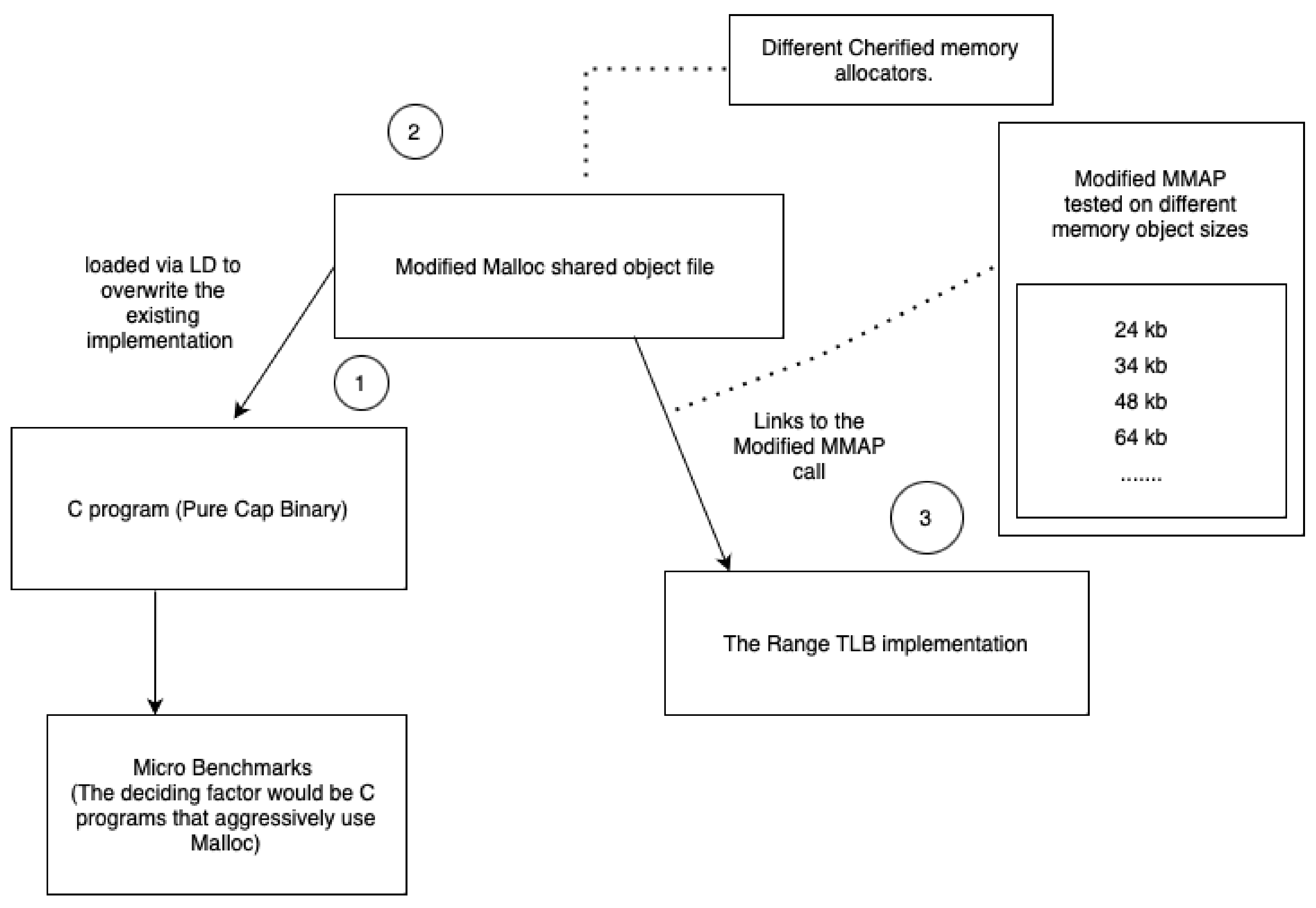
The CHERI hardware architecture is designed to provide security guranetees, This has had performance implications such as more L1 TLB misses on purecap mode rather than running an application as a 64 bit program. The following expirement proposes on adding range TLB which is stored on the page table to store TLB translation entries of large memory objects. During this process the range ID would be added to the pointer which allows looking up the TLB entries at an earlier stage during the address generation stage of the pipeline.

Research Questions

Research Questions:

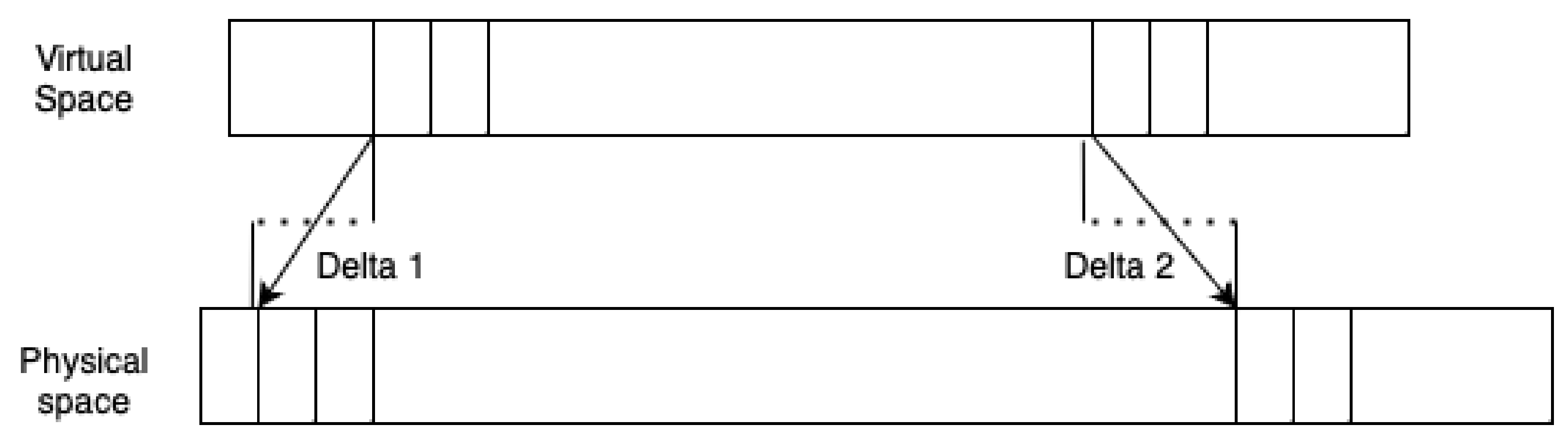
- (RQ1) It is possible to use the unused meta-data field in FAT-Pointers to reduce the number of TLB misses to reduce the CPU Stall time ?
- (RQ2) Can the number of L1 DTLB misses be minised by offloading translation lookups to the newly introduced range TLB ?

Expirement Proposal

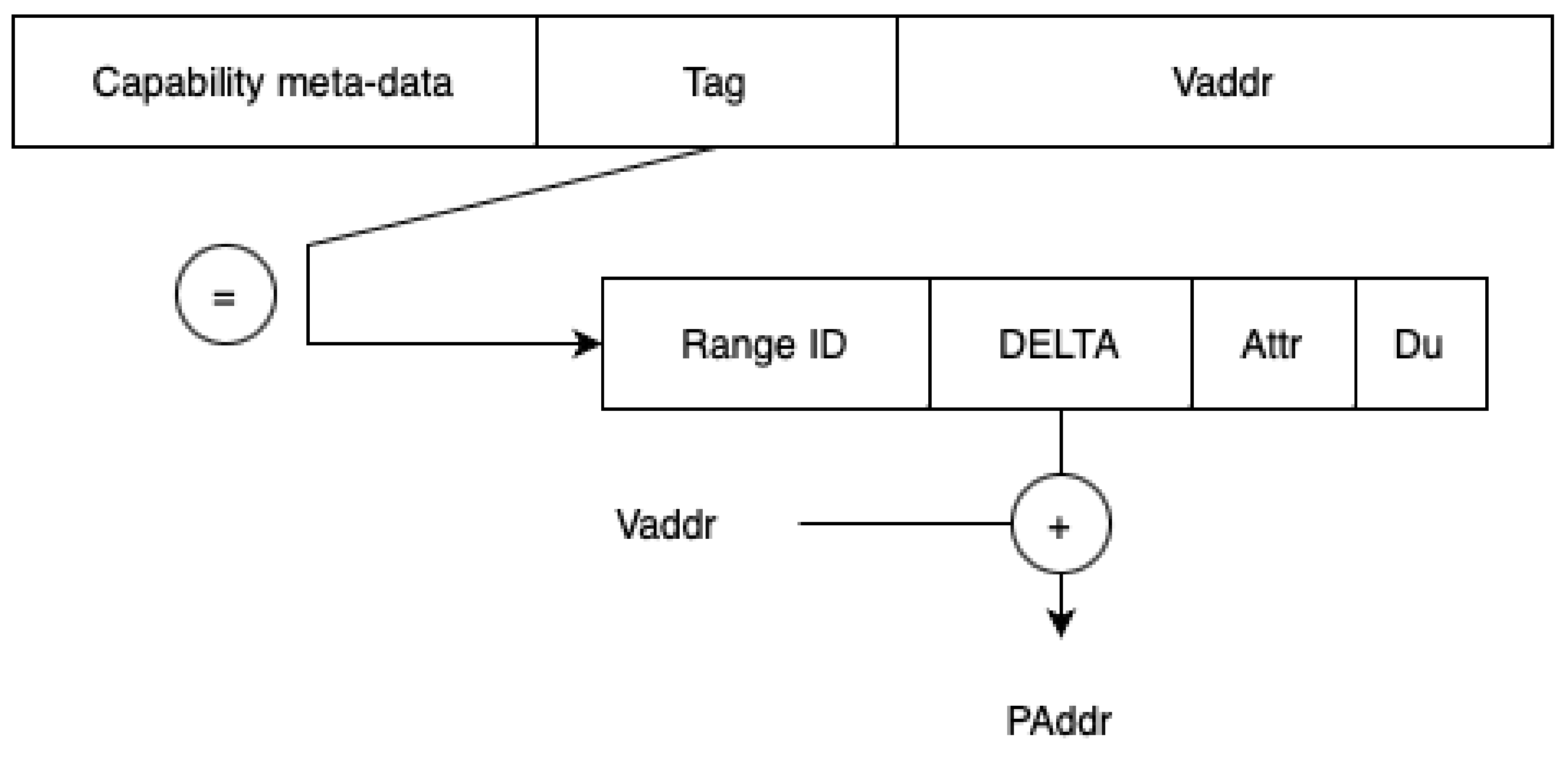

Different Cherified memory allocators.
2
Modified Malloc shared object file
loaded via LD to overwrite the existing implementation
1
C program (Pure Cap Binary)
Micro Benchmarks (The deciding factor would be C programs that aggressively use Malloc)
Modified MMAP tested on different memory object sizes
24 kb
34 kb
48 kb
64 kb
....
Links to the Modified MMAP call
3
The Range TLB implementation

- *A set of Microbenchmarks which are designed to create pressure on the L1 TLB to create more L1 misses.*
- *Modified Malloc to call the new syscall mmap_range for large memory objects.*
- *Testing different memory object sizes to use the range TLB.*

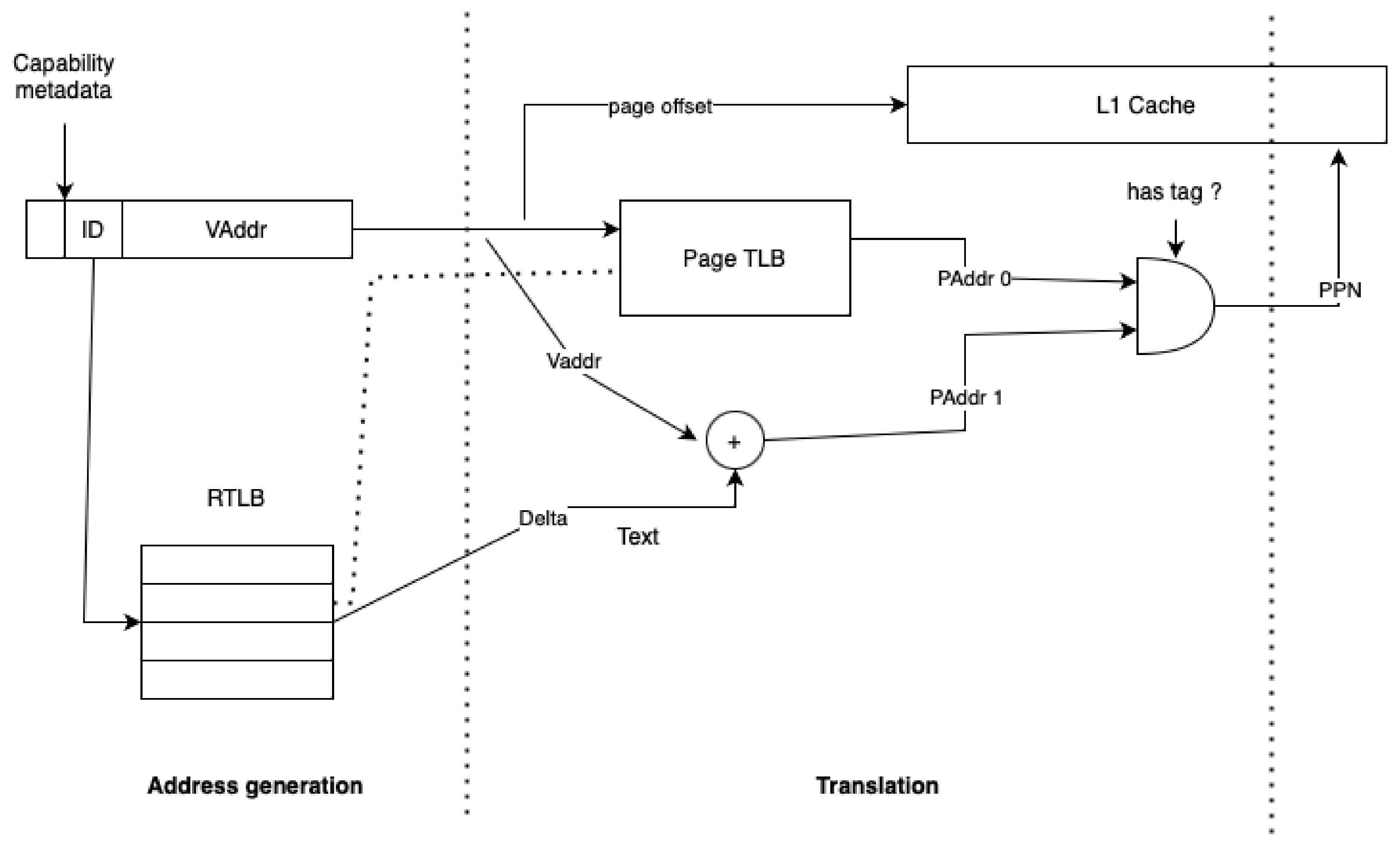
Range TLB



The current expirement adopts the range translation concept from FlexPointer[1] which was inturn adopted from RMM[2]. A range consists of contigous virtual and phyiscal pages. Addresses share a common Delta. To assume physical contiguity of a particular range we are proposing to use eager paging strategy as proposed from FlexPointer[1].



Range TLB is organised as a fully associative cache, As per the conventional the page table the lower 12 bits is used record attributes (eg: R/W: Read/Write).



Resources